Blocks: Creating Rich Tables with Drag-and-Drop Interaction

Allison Whilden, Dirk Karis, Vidya Setlur, Rodion Degtyar, Jonathan Que, and Filippos Lymperopoulos

Tableau Software

Abstract

We present Blocks, a formalism that enables the building of visualizations by specifying layout, data relationships, and level of detail (LOD) for specific portions of the visualization. Users can create and manipulate Blocks on a canvas interface through drag-and-drop interaction, controlling the LOD of the data attributes for tabular style visualizations. We conducted a user study to compare how 24 participants employ Blocks and Tableau to complete a target visualization task. Findings from the study suggest that Blocks is a useful mechanism for creating visualizations with embedded microcharts, conditional formatting, and custom layouts. We describe future directions for extending Blocks in visual analysis interfaces.

CCS Concepts

• ***Human-centered computing*** → ***Information visualization;***

1 Introduction

Visual analysis tools help support the user in data exploration and iterative view refinement. Some of these tools are more expressive, i.e., giving expert users more control, while others are easier to learn and faster to create visualizations. These tools are often driven by underlying grammars of graphics [Wil05] that provide various formalisms to concisely describe the parts of a visualization. Reasonable defaults are applied to infer missing information to generate a valid graphic. The downside of these concise representations is that the support for expressiveness for visualization generation in these tools is either limited or difficult for a user to learn how to do.

Drag-and-drop is one paradigm for supporting task expression through user interaction where the visibility of the object of interest replaces complex language syntax. VizQL [STH02] is one such formalism that supports the expression of chart creation through direct manipulation. While the language enables users to create charts through its compositional algebra, there is still a tight coupling between the query, the visualization, and the layout. Users often spend significant time generating complex visualizations when they have a specific structure in mind. The other paradigm for promoting expressiveness for chart creation is through the programmatic use of declarative specification grammars [Bos12, SRHH16, SMWH17].

Despite the prevalence of these tools, creating expressive data visualizations still remains a challenging task. Beyond having insight about how the data can be best visualized, users need to have sufficient knowledge in using these tools for generating visualizations. So, how can we support users in their analytical workflows by enabling a greater degree of flexibility and control over nesting relationships, layout, and encodings yet providing the intuitiveness of a user interface? In this paper, we explore this dichotomy between expressibility and ease of use when creating rich tables.

Specifically, our contributions are as follows:

- We introduce *Blocks*, a formalism that builds upon VizQL by supporting the nested relationships between attributes in a visualization using a drag-and-drop interaction. Every component of the visualization is an analytical entity to which different nesting and encoding properties can be applied.
- We implement a *Blocks System* that provides a user increased flexibility with layout and formatting options through the direct manipulation of Block objects in the interface.
- We conducted a user study with 24 participants involving rich tables using Tableau and Blocks. Findings from the studies indicate that Blocks is a promising paradigm for the creation of rich tables and opens up additional research directions to pursue.

2 Related Work

2.1 Declarative specification grammars

Declarative visualization languages address the problem of expressiveness by allowing developers to express how they would like to render a visualization. Vega [SRHH16] and Vega-Lite [SMWH17] support the authoring of interactivity in the visualizations. While these specification languages provide a great degree of flexibility in how charts can be programmatically generated, they provide limited support for displaying different levels of granularity within a field in a visualization. Viser [WFB*19] addresses this gap by automatically synthesizing visualization scripts from simple visual sketches provided by the user. Specifically, given an input data set and a visual sketch that demonstrates how to visualize a very small subset of

this data, their technique automatically generates a program that can be used to visualize the entire data set. Ivy [MC21] proposes parameterized declarative templates, an abstraction mechanism over JSON-based visualization grammars. Harper and Agrawala [HA16] convert D3 charts into reusable Vega-Lite templates. While our work is similar to that of declarative grammars and template specifications for abstracting low-level implementation details from the user, we specifically support nested queries, layout, and encoding flexibility through drag-and-drop interaction in the Blocks interface.

2.2 Visual analysis interfaces

Visual analysis tools have developed ways to help users in getting started. The basic form of these tools for chart generation include chart pickers [Ahl96]. Tableau and PowerBI, along with systems like Charticulator [RLB19] are built on a visualization framework that enables users to map fields to visual attributes using drag-and-drop interaction. As more analytical capabilities are enabled in these tools, there is a disconnect from the underlying abstraction, leading to calculation editors and dialog menus that add both complexity and friction to the analytical workflow.

Mixed-initiative VisRec systems combine manual specification with recommendations [HOH18, KHPA12, YEB18, vdEvW13, LBW19, Goo15, VWS*18, LMH20]. These systems however, offer little or no suggestions for how to create visualizations that are more complex in terms of nesting or encoding properties. Prior work has also explored combinations of interaction modalities for creating visualizations. For example, responsive matrix cells combine focus and context with semantic zooming to allow analysts to go from the overview of the matrix to details in cells [HBS*21]. Liger [SJPE19] combines shelf-based chart specification and visualization by demonstration. Hanpuku [BDFM17], Data-Driven Guides [KSL*17], and Data Illustrator [LTW*18] combine visual editor-style manipulation with chart specification. Domino [GGL*14] is a system where users can arrange and manipulate subsets, visualize data, and explicitly represent the relationships between these subsets. However, Domino has limited nesting and inheritance capabilities as it does not define parent-child relationships between Blocks. Our work specifically addresses this gap and focuses on enabling more expressive charts with nesting by using drag-and-drop as an interaction paradigm.

3 Design Goals

To better understand the limitations for creating more expressive visualizations, we interviewed 19 customers, analyzed 7 internal dashboards, and reviewed 10 discussions on the Tableau Community Forums [tab21a] that used various workarounds to accomplish their analytical needs. From this analysis, we identify three design goals:

- **DG1. Support drag-and-drop interaction**: Our goal is to maintain the ease of use provided by the drag-and-drop interface and data-driven flow when creating visualizations.
- **DG2. Better control over visualization components and layout**: When users have specific ideas of what they want to create, their workflows often conflict with the system defaults. Our goal is to support users with increased layout flexibility as they generate charts for their analytical needs.
- **DG3. Aggregate and encode at any Level of Detail (LOD) in a visualization**: Our goal is to provide the ability to evaluate data attributes aggregated at any LOD in the visualization.

4 The Blocks Formalism

The Blocks formalism employs a set of connected local expressions (i.e., *Blocks*). Each Block represents a single query and builds a component of the visualization from it. Parent-child relationships between the Blocks form a directed acyclic graph (DAG).

Any `field-instance` without an `aggregation` is called a Dimension; the set of all Dimensions in a given query is called the LOD of that query. Fields with an aggregation are called Measures; they are aggregated within groups defined by the LOD. The *local LOD* of the Block is the set of all dimensions used by any `encoding` within the Block. The *full LOD* of the Block is the union of its local LOD and that of all of its ancestors. All of the measures used by the Block are evaluated at the full LOD of the Block. In addition to defining the LOD, the `encodings` map the query results to visual and spatial encodings. By providing a means to encode X (x-axis) and Y (y-axis) for each visualization component, Blocks addresses **DG3** with respect to sparklines and other micro charts.

```
block          := (block-name, layout-type,
                   mark-type, encoding, children)
children       := {(child-group)}
child-group    := {block-name}
layout-type    := "rows" | "columns" | "inline"
mark-type      := "text" | "shape" | "circle"
                | "line" | "bar"
encoding       := ({encoding-type},
                   field-instance)
encoding-type  := "color" | "size"
                | "shape"
                | "text" | "x-axis"
                | "y-axis" | "sort-asc"
                | "sort-desc" | "detail"
field-instance := ([aggregation], field-name)
```

Each Block renders one mark of its `mark-type` per tuple in its query result. The `layout-type` determines how each of the Block's rendered marks is laid out in space. A Block with the layout type of `rows` (`columns`) creates a row (column) for each value in its domain, each containing a single mark. A Block with the layout type of `inline` renders all of its marks in a single shared space. Child Blocks are laid out in relation to their parents' positioning. A `child-group` is a set of children that share the same row (for a `rows` parent) or column (for a `columns` parent).

5 The Blocks System

The Blocks system provides an interface for creating Blocks and viewing the resulting visualizations, as shown in Figure 1.

5.1 Blocks interface

The Blocks interface provides a drag-and-drop technique to encode fields, consistent with **DG1**. Pills represent fields and a schema

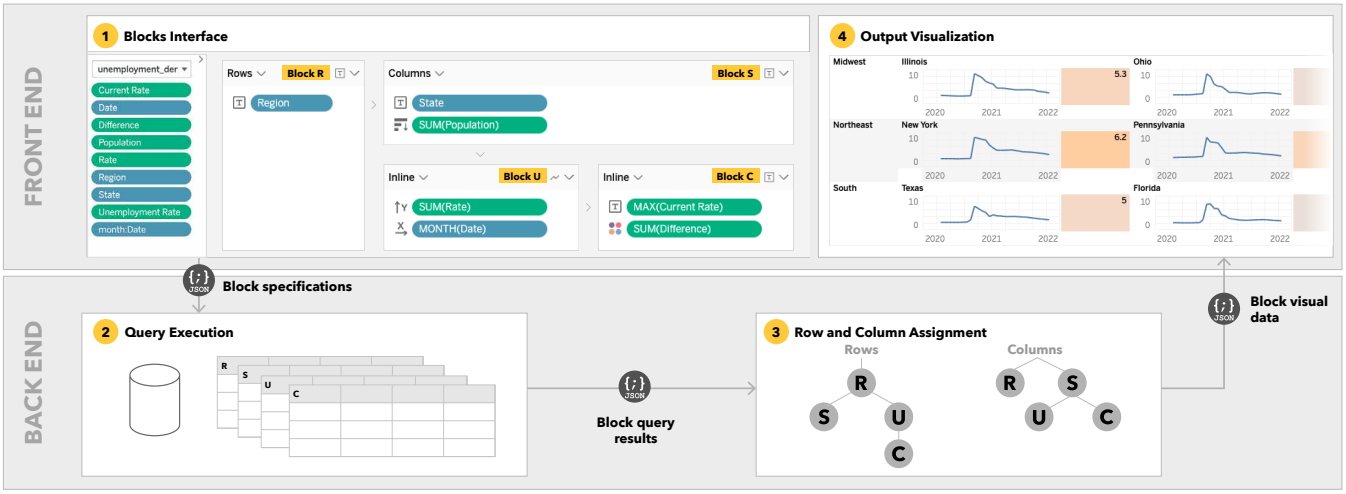

Figure 1: *Blocks system overview. Users create* Block GUI Cards *shown in the interface (1), that define multiple field encodings at a single LOD. The* Block GUI card *is translated into a* Block specification*. The specification consists of dimensions, measures aggregated to the LOD of the cross product of the dimensions, layout, visual encodings, mark type, filters, and sort order. From this* Block specification*, a* Block query *is issued to the data source (2). The output of a* Block query *is a* Block result set *with row and column assignments (3) that returns the tuples and corresponding encoding results, rendered as an* output visualization *(4).*

pane contains the list of fields from the data source. The Blocks interface provides a canvas that supports an arbitrary number of Blocks. Dragging out a pill to the canvas will create a new Block, defaulting the Block's encoding, mark type, and layout type based on metadata of the field that the pill represents. For example, dragging out a pill that represents a discrete string field will create a Block with the layout type of rows, mark type of text, and that field encoded on T . The layout type and mark type are displayed at the top of the Block. Encodings are displayed as a list inside the Block. Additional pills can be dragged to the canvas to create a new, unrelated Block, added as an additional encoding to an existing Block, or dropped adjacent a Block to create a new related Block.

When a pill is dragged over an existing Block, drop targets appear that represent any unused encodings in that Block that the system provides. When a pill is dragged over an area adjacent to an existing Block, drop targets appear to assist in creating a new related Block. Blocks placed to the right of or below related Blocks are automatically determined to be child Blocks, shown by a chevron icon (›). Every Block must have both a Rows and a Columns parent to determine its position in the visualization; more than two parents are not permitted. If a Rows or Columns parent is not explicit in the interface, that parent is added implicitly by the system. The leftmost and topmost Blocks are considered children of implicit Rows root and Columns root Blocks. A Block that has a Rows parent but not a Columns parent uses the Column parent of its Rows parent, and the converse for one with only a Columns parent. Inline Blocks do not have children. If a Block appears in the interface as a child of an Inline Block, it instead uses the Rows and Columns parents of the Inline Block. Figure 1 (Section 1) shows the Blocks interface with a Rows Block, *R* and a Columns Block child, *S*. Block *S* has two inline Block children, one of which, Block *U*, shows a sparkline, while Block *C*, shows a single value with a color encoding (**DG2**).

5.2 Row and Column Assignment

Each Block executes a single query resulting in multiple tables with different schemas. The system assigns Row and Column indexes from a single grid to tuples from all of these tables. This section describes the process for Rows; it is repeated for Columns.

1. Produce a Block tree from the Blocks DAG by only considering links from Rows Blocks to their children, excluding any other links. The Blocks Interface ensures that this tree exists, is connected, and has a single root at the implicit Rows root Block. Figure 1 (Section 3) shows the trees for this visualization. Note the implicitly-added parents, e.g., Block U is assigned R as its Rows parent by way of Block S.
2. Produce a tuples tree by treating each tuple as a node. Its parent is the tuple from its parent Block with matching dimension values.
3. Sort the children of each tuple, first in the order their Blocks appear as children in the Blocks tree, and then in the order of the Rows dimensions and user-specified sorts, if any, for each Block.
4. Assign row indexes to each tuple by walking the tuple tree in depth-first order. Leaf tuples get a single, sequentially assigned row index; interior nodes record the minimum and maximum row indexes of their leaves.

5.3 Output visualization

Each tuple from a Rows or Columns Block forms a single cell containing a single mark. Tuples from an Inline Block with the same Row and Column parent tuples form a single cell. The values

of visual encoding fields that are dimensions differentiate between marks within that cell. Each cell is rendered on the grid at the row and column indexes assigned by the previous step. Marks may comprise a visualization (e.g., bar chart, scatter plot) depending on visual encodings of the Block.

6 User Study

We conducted a user study to understand how Blocks could be useful and how the paradigm could integrate into a more comprehensive visual analysis system. 24 volunteer participants (6 females, 18 males) took part, recruited from a visual analytics organization. Participants had a variety of backgrounds - user researcher, sales consultant, data analyst, product manager, and technical program manager. Eight users were experienced, eight had moderate experience, while eight had limited proficiency with visual analytics tools.

6.1 Procedure and Apparatus

Two of the authors supported each session, one being the facilitator and the other as the notetaker. The study trials were done remotely over shared screen video conferences. All sessions took approximately 50 minutes and were recorded. We began the study with the facilitator reading from an instructions script, followed by a two-minute tutorial video of Blocks. Experimenter script, task instructions, and tutorial video are included in supplementary material.

Part 1: Open-ended exploration This task enabled us to observe how people would explore the Blocks interface using the Superstore dataset [sup21]. Instructions were: "Based on what you saw in the tutorial video, we would like you to explore this data in the Blocks prototype. Please let us know what questions or hypotheses you're trying to answer while using the interface."

Part 2: Closed-ended tasks The closed-ended tasks were intended to provide some consistent objectives for task comparison across both Tableau and Blocks systems. Participants completed one of three randomly assigned closed-ended tasks with corresponding datasets - Creating a cross tab with barcharts [tit20], creating and sorting a rich table [Gap20], or a table with sparklines [cov20]. Participants performed the assigned task with a Tableau Online [tab21b] workbook and the Blocks prototype.

We conducted a thematic analysis through open-coding of session videos, focusing on strategies participants took. We use the notation *P#* to refer to the study participants. We also conducted two-week long diary studies with eight participants and include their task flows in the supplementary material of the paper.

7 Study Findings

All participants were able to complete the three tasks in Blocks, while three participants needed guidance to sort a table and add sparklines in Tableau. To understand how intuitive the Blocks paradigm is for users, we first examine the strategies participants adopted for sense-making within the interface. Participants immediately dragged attribute pills from the data pane onto the canvas. P4 remarked, "I'm going to drag `Category` on the canvas, and I see that it created a Block showing the various category values." Participants often dragged around Blocks to change the structure of the visualization. The drop targets around a Block piqued participants' curiosity in exploring what would happen when they dragged out pills to these targets. P8 remarked, "I'd like to get an intuitive sense as to what happens when I drop it here [pointing below the Block] or there [pointing to the right]." Participants also found it useful to modify the existing chart by dragging pills into new Blocks in the middle of or adjacent to other Blocks, breaking down attributes into targeted levels of detail. They found the visual layout of the Blocks to directly inform the structure of the generated chart – "The LOD of what is to the right is defined by what is to the left [P2]" and "You build out the viz literally the way you think about it [P6]."

8 Discussion

General feedback from the participants was positive and suggested that Blocks is a promising paradigm to have more control over the layout and manipulating the LOD in the visualization. P12 remarked, "This is ridiculously awesome. I'm not going to lie, but I have this horrific cross tab bookmarked to do in Tableau. I can see doing it in Blocks in a minute and a half." Participants appreciated the flexibility of being able to apply conditional formatting to different parts of a visualization. P19 commented, "That's cool. I've never been able to do conditional color dimensions before." Having more control over LOD was a consistent feature that participants found useful. P6 said, "Aha! I can get sparklines so easily."

There were limitations with the Blocks prototype. The flexibility that the Blocks interface affords comes with an inherent downside of a vast set of drop-target options. P10 was overwhelmed with the choices when he initially started exploring and remarked, "There are so many arrows to choose from. It would be helpful if I can get a hint as to where I should drop by pill based on what attribute I selected." Showing previews in the interface could better orient the user to the workings of the interface. P12 suggested, "It would be really cool if there are actions associated with the visual indicators of the drop targets so the interface does not feel too free form." Participants wanted customization in the interface. P3 said, "It would be nice if I could center the sparklines to the text in the table. I would also like to add a dot on the maximum values in the sparklines." Participants also wanted advanced analytical capabilities such as adding calculated fields to the visual panes in the charts.

9 Conclusion

We present Blocks, a new formalism that builds upon VizQL by supporting the handling of nesting relationships between attributes through direct manipulation. By treating each component of the visualization as an analytical entity, users can set different LOD and encoding properties through drag-and-drop interactions in the Blocks interface. An evaluation of the Blocks interface and comparing users' analytical workflows with Tableau indicates that Blocks is a useful paradigm for supporting the creation of rich tables. Future research directions will explore additional analytical and interaction capabilities in the system along with useful scaffolds for supporting users during visual analysis. We hope that the insights from our work can help strike a balance between expressivity, ease of use, and analytical richness in visual analysis tools.

References

[Ahl96] AHLBERG C.: Spotfire: An information exploration environment. *SIGMOD Rec. 25*, 4 (Dec. 1996), 25–29. URL: https://doi.org/10.1145/245882.245893, doi:10.1145/245882.245893. 2

[BDFM17] BIGELOW A., DRUCKER S., FISHER D., MEYER M.: Iterating between tools to create and edit visualizations. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 481–490. doi:10.1109/TVCG.2016.2598609. 2

[Bos12] BOSTOCK M.: D3.js - data-driven documents. http://d3js.org/. 1

[cov20] Covid-19 Dataset, 2020. CC-BY Dataset: https://covid19.ca.gov. 4

[Gap20] GAPMINDER: World development indicators, 2020. CC-BY Dataset: https://gapminder.org/data. 4

[GGL*14] GRATZL S., GEHLENBORG N., LEX A., PFISTER H., STREIT M.: Domino: Extracting, comparing, and manipulating subsets across multiple tabular datasets. *IEEE Transactions on Visualization and Computer Graphics (InfoVis) 20*, 12 (2014), 2023–2032. doi:10.1109/TVCG.2014.2346260. 2

[Goo15] GOOGLE: Explore in Google Sheets. https://www.youtube.com/watch?v=9TiXR5wwqPs. 2

[HA16] HARPER J., AGRAWALA M.: Converting basic d3 charts into reusable style templates. *IEEE Transactions on Visualization and Computer Graphics PP* (09 2016). doi:10.1109/TVCG.2017.2659744. 2

[HBS*21] HORAK T., BERGER P., SCHUMANN H., DACHSELT R., TOMINSKI C.: Responsive matrix cells: A focus+context approach for exploring and editing multivariate graphs. *IEEE Transactions on Visualization and Computer Graphics 27* (2021), 1644–1654. 2

[HOH18] HU K., ORGHIAN D., HIDALGO C.: Dive: A mixed-initiative system supporting integrated data exploration workflows. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics* (2018), ACM, p. 5. 2

[KHPA12] KEY A., HOWE B., PERRY D., ARAGON C.: VizDeck. *Proceedings of the 2012 international conference on Management of Data - SIGMOD '12* (2012), 681. URL: http://dl.acm.org/citation.cfm?doid=2213836.2213931, doi:10.1145/2213836.2213931. 2

[KSL*17] KIM N. W., SCHWEICKART E., LIU Z., DONTCHEVA M., LI W., POPOVIC J., PFISTER H.: Data-driven guides: Supporting expressive design for information graphics. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 491–500. doi:10.1109/TVCG.2016.2598620. 2

[LBW19] LAW P.-M., BASOLE R. C., WU Y.: Duet: Helping data analysis novices conduct pairwise comparisons by minimal specification. *IEEE transactions on visualization and computer graphics 25*, 1 (2019), 427–437. 2

[LMH20] LIN H., MORITZ D., HEER J.: Dziban : Balancing Agency & Automation in Visualization Design via Anchored Recommendations. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems - CHI '20* (2020). 2

[LTW*18] LIU Z., THOMPSON J., WILSON A., DONTCHEVA M., DELOREY J., GRIGG S., KERR B., STASKO J.: Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. 1–13. URL: https://doi.org/10.1145/3173574.3173697. 2

[MC21] MCNUTT A. M., CHUGH R.: Integrated visualization editing via parameterized declarative templates. *ArXiv abs/2101.07902* (2021). 2

[RLB19] REN D., LEE B., BREHMER M.: Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (Jan. 2019), 789–799. URL: https://doi.org/10.1109/TVCG.2018.2865158, doi:10.1109/TVCG.2018.2865158. 2

[SJPE19] SAKET B., JIANG L., PERIN C., ENDERT A.: Liger: Combining interaction paradigms for visual analysis, 2019. arXiv:1907.08345. 2

[SMWH17] SATYANARAYAN A., MORITZ D., WONGSUPHASAWAT K., HEER J.: Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan. 2017), 341–350. doi:10.1109/TVCG.2016.2599030. 1

[SRHH16] SATYANARAYAN A., RUSSELL R., HOFFSWELL J., HEER J.: Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2016). URL: http://idl.cs.washington.edu/papers/reactive-vega-architecture. 1

[STH02] STOLTE C., TANG D., HANRAHAN P.: Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (Jan. 2002), 52–65. URL: https://doi.org/10.1109/2945.981851, doi:10.1109/2945.981851. 1

[sup21] Tableau Superstore, 2021. CC-BY Dataset: https://help.tableau.com/current/guides/get-started-tutorial/en-us/get-started-tutorial-connect.htm. 4

[tab21a] Tableau Community Forum. https://community.tableau.com, 2021. 2

[tab21b] Tableau Online, 2021. https://online.tableau.com. 4

[tit20] Encyclopedia Titanica, 2020. CC-BY Dataset: https://www.encyclopedia-titanica.org. 4

[vdEvW13] VAN DEN ELZEN S., VAN WIJK J. J.: Small multiples, large singles: A new approach for visual data exploration. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 191–200. 2

[VWS*18] VIEGAS F., WATTENBERG M., SMILKOV D., WEXLER J., GUNDRUM D.: Generating charts from data in a data table. *US 20180088753 A1* (2018). 2

[WFB*19] WANG C., FENG Y., BODIK R., CHEUNG A., DILLIG I.: Visualization by example. *Proc. ACM Program. Lang. 4*, POPL (Dec. 2019). URL: https://doi.org/10.1145/3371117, doi:10.1145/3371117. 1

[Wil05] WILKINSON L.: *The Grammar of Graphics (Statistics and Computing)*. Springer-Verlag, Berlin, Heidelberg, 2005. 1

[YEB18] YALÇIN M. A., ELMQVIST N., BEDERSON B. B.: Keshif: Rapid and expressive tabular data exploration for novices. *IEEE transactions on visualization and computer graphics 24*, 8 (2018), 2339–2352. 2